Softmax Regression

Hoàng Vũ Minh

Softmax regression (hay còn gọi là multinomial logistic regression) mở rộng từ logistic regression (sử dụng cho bài toán phân loại nhị phân) để có thể xử lý các bài toán phân loại với nhiều lớp. Ý tưởng chính là biến đổi tổng quát các đầu ra tuyến tính thành xác suất cho từng lớp.

Ta hình dung đơn giản: với nhiều class, ta cần đầu ra của chúng vẫn là xác suất, tuy nhiên tổng các xác suất đó phải bằng 1. Điều này dẫn đến việc cần sử dụng một hàm kích hoạt khác thỏa mãn điều kiện: nó vẫn chuyển đầu ra của hqtt thành xác suất, tuy nhiên tổng các xác suất của các lớp =1. Đó chính là hàm softmax

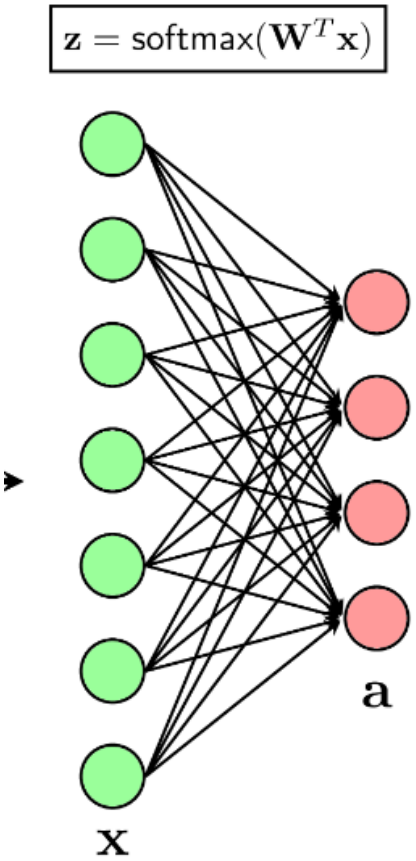
1. Hàm softmax:

$$a_i = \frac{\exp(z_i)}{\sum_{j=1}^{C} \exp(z_j)}, \quad \forall i = 1, 2, \ldots, C$$

Lý do sử dụng hàm Softmax:

- Convert các giá trị về trông khoảng 0 1, phù hợp với xác suất, tổng kết quả softmax ở các lớp =1 nên phù hợp với việc tính xác suất cho nhiều nhãn.
- Một điểm hay của softmax là nó sử dụng hàm số mũ. hàm số mũ giúp ***khuếch đại giá trị đầu vào*** một cách rất nhanh, tức là nó sẽ tạo ra sự chênh lệch ở đầu ra giữa các giá trị, thể hiện rõ hơn về mặt xác suất cho kết quả đúng.
- Tuy nhiên vấn đề ở đây, như đã trình bày ở logistic regression, exp(z) sẽ ***tăng cực nhanh nếu như z lớn***. Chính vì vậy, giả sử trong bài toán xuất hiện một zi nào đó rất lớn, kết quả sẽ bị tràn số. Vì vậy, người ta thường xem xét ***trừ tất cả*** các z trong hàm softmax đi ***một hằng số c*** nào đó để tránh hiện tượng này. Và vì softmax là một hàm đồng biến nên việc trừ đi không ảnh hưởng đến thứ tự độ lớn của các kqua.

Công thức hàm này rất đơn giản và nó cũng thỏa mãn những điều đã đặt ra ở trên. Việc chia cho tổng sẽ làm cho tổng các xác suất bằng 1 đồng thời giúp cho giá trị luôn nằm trong khoảng 0 1.



Minh hoạ hoạt động của softmax theo kiểu neural network. Ta có thể ngầm hiểu lớp a bao gồm cả lớp z, tức là hàm softmax ở đây được áp dụng để convert đầu ra từ layer gần cuối thành xác suất cho 4 lớp màu đỏ được mô tả trong hình. Trong các mạng deep learning, Softmax cũng thường được đặt làm layer cuối cùng nhằm mục đích chuyển các feature thành xác suất các lớp cho các bài toán classification.

2. *Hàm mất mát*

Hàm mất mát: Softmax cũng sử dụng Cross Entropy để đo lường sự chênh lệch xác suất dự đoán và nhãn thực tế.

Công thức Cross-Entropy giữa 2 giá trị xác suất:

$$H(\mathbf{p}, \mathbf{q}) = -\sum_{i=1}^{C} p_i \log q_i$$

Ta có thể thấy, khi p ở xa q, loss rất cao, đồng nghĩa với việc điểm phạt của nó cho sự

Công thức loss function được sử dụng trong Softmax (vẫn là cross entropy, chỉ là thay đổi và thêm các ký hiệu):

$$J(\mathbf{W}; \mathbf{x}_i, \mathbf{y}_i) = -\sum_{j=1}^{C} y_{ji} \log(a_{ji})$$

Công thức này là cho 1 điểm dữ liệu (dùng chỉ số i), nếu tổng loss thì thêm xích ma N với N là số sample nữa. Thật ra công thức này "liêm" hơn với khái niệm của loss function (vì loss function thường là đề cập cho mức độ data point), còn tổng thì là coss function.

Giải thích thêm về cross-entropy cho multi-class:

Nhãn của nó sẽ như kiểu one-hot coding, tức là vdu xét sample i trong bài toán 3 lớp A, B, C, nhãn thật của nó là 1

- Tức là y tại mẫu này = [1, 0, 0]
- giả sử a tại mẫu này = [0.8, 0.15, 0.05]
- Loss: - y1 log a1 -y2 log a2 - y3 log a3

= -y1 log a1 = -log a1

Lý do là vì ở các vị trí khác y của nó bằng 0, nên không cần quan tâm nữa. Tức là nhìn chung ở nhãn nào thì ta chỉ tính chênh lệch ở cái xác suất rơi vào nhãn đó thôi.